Carleton University
Department of Systems and Computer Engineering
SYSC 1005 - Introduction to Software Development - Fall 2012

Lab 10 - Lists, Tuples, Sets and Files

Attendance/Demo

To receive credit for this lab, you must make an effort to complete the exercises, and demonstrate your work.

When you have finished all the exercises, call a TA, who will review the code you wrote. For those who don't finish early, a TA will ask you to demonstrate whatever code you've completed, starting about 30 minutes before the end of the lab period. Finish any exercises that you don't complete by the end of the lab on your own time.

Part 1 - Maintaining A Club Membership List

Suppose we want to store information about the members of a club. Individual memberships can be represented as a tuple containing three values: a string containing the name of the member, an integer representing the month in which the member joined the club (1 = January, 2 = February, etc.), and an integer representing the year in which the member joined. For example, here is the tuple for a club member who joined in February, 2011: `("Jack Harkness", 2, 2011)`.

The membership list can be maintained as a list of these tuples.

Exercise 1

Open a new file in Wing IDE 101, and save it with the name `club.py`.

Define a function named `number_of_members` that is passed a membership list. This function will return the current size of the club's membership. The function header is:

```
def number_of_members(members):
```

Parameter `members` is a Python list.

Type the following statements in the Python shell to verify that the function returns 0 when it is passed an empty membership list.:

```
>>> members = []
>>> number_of_members(members)
0
```

Until you've defined a function that adds memberships to the list, `number_of_members` will always return 0, but it will be ready for further testing later.

Exercise 2

Define a function named `join`, which adds a new member to the club's membership list. The function header is:

```
def join(members, name, month, year):
```

- members is the membership list (a Python list)
- name is the name of the member (a string)
- month is the month in which the member joined (an integer between 1 and 12)
- year is the year in which the member joined (an integer)

Remember that the information about an individual member is to be stored in a tuple.

This function does not return anything. Instead, it modifies the list bound to members.

If the month parameter is outside the valid range of 1-12, the function should not change the membership list, but should instead print a descriptive error message.

Use the shell to test your function. Create an empty membership list, and call join a few times to add members to the list. Each time you call join, verify that the new member was added to the list. Also, call number_of_members and verify that it correctly returns the new size of the club's membership. For example:

```
>>> members = []
>>> join(members, "Gwen", 4, 2010)
>>> members     # the membership list will be displayed
>>> number_of_members(members)
```

Exercise 3

Define a function named build_club, which returns a membership list containing the membership information for a small club. The function header is:

```
def build_club():
```

This function must call your join function from Exercise 2 to add each member to the membership list. The function should return the list shown in this example:

```
>>> members = build_club()
>>> members
[("Jack", 2, 2011), ("Ianto", 2, 2009), ("Gwen", 4, 2010), ("Rhys", 5, 2008),
("Owen", 2, 2010), ("Toshiko", 3, 2011)]
```

Exercise 4

Define a function named joined_in_month that determines how many members joined in a given month. The function header is:

```
def joined_in_month(members, month):
```

- members is the membership list (a Python list)
- month is the month in which the member joined (an integer between 1 and 12)

This function returns the number of members who joined in the month we are interested in.

If the month parameter is outside the valid range of 1-12, the function should print a descriptive error message and return -1.

Use the shell to test your function. Use the membership list returned by your build_club function. You'll need several test cases; e.g., develop tests in which joined_in_month returns 0 (no members joined in the specified month), 1 (one member joined in the specified month), and a value greater than 1 (multiple members joined in the specified month). You'll also need test cases to demonstrate the your function correctly handles out-of-range values for the month parameter.

Exercise 5

Define a function named purge that removes from the club's membership list the membership information for all members who joined in a given month, and returns this information in a separate list. The function header is:

```
def purge(members, month):
```

- members is the membership list (a Python list)
- month is the month in which the member joined (an integer between 1 and 12)

The function returns a new list containing the membership information for every member who joined in the given month; that is, all the tuples that were removed from members.

If the month parameter is outside the valid range of 1-12, the function should print a descriptive error message and return an empty list.

Hint: before attempting this exercise, you should review the solutions to the remove_selected challenge exercise from Lab 8. (These solutions were presented in a lecture, and are posted on cuLearn.) The two functions are similar, in that both functions must modify a list while iterating over it. The purge function will modify the membership list (remove tuples, causing the remaining tuples to "shift to the left") while it traverses the list. Remember to take this into account while designing your function.

Use the shell to test your function. Use the membership list returned by your build_club function. You'll need several test cases; e.g., develop cases where purge returns an empty list (no members joined in the specified month), a list of size 1 (one member joined in the specified month), and a list whose size is greater than 1 (several members joined in the specified month). For each test case, verify that membership list passed to purge is modified correctly. You'll also need test cases to demonstrate the your function correctly handles out-of-range values for the month parameter.

Part 2 - Challenge: Listing the Words in a File

Open a new file in Wing IDE 101, and save it with the name word_list.py.

Define a function named build_word_list that is passed a string containing the name of a text file. The function header is:

```
def build_word_list(filename):
```

The function will first open the specified file for reading. It will then build and return a list of all the words in the text file, such that:

- There are no duplicate words in the list. For example, if the file contains "and" five times, "and" will appear in the list only once.
- All words must be converted to lower case.
- Any punctuation characters (e.g., commas, periods, question marks, etc.) that occur immediately before or after words are removed. For example, the string "Hello, world! " would result in the words "hello" and "world" being added to the list (assuming they aren't already in the list).
- The words must be sorted into ascending order.

This function is fairly easy to write if you use lists and/or tuples and/or sets. It's up to you to decide which collections are the best ones to use, but **do not** use Python dictionaries (values of type dict). Try not to reinvent the wheel; instead, use methods provided by the collection types whenever possible. You can find out more about these methods by using Python's help facility:

```
>>> help(str)
>>> help(list)
>>> help(tuple)
>>> help(set)
```

I strongly suggest you use an iterative, incremental approach, instead of trying to define and debug the entire function in one step. For example, you could start by writing a function that reads lines from the text file one at a time, and as each line is read, gets all the words from the line and adds them in a list. Don't worry about handling punctuation characters, removing duplicate words, converting the words to lower case or sorting the list. Test this function with a short text file. After you're convinced that this version of the function is correct, modify it to do a bit more; for example, handle punctuation marks, then retest the function. Continue working this way, adding a bit more functionality with each iteration.

When you've finished your function, test it using the file sons_of_martha.txt.

A Note about Rudyard Kipling and *The Sons of Martha*

Rudyard Kipling (1865 - 1936) was an English poet, short-story writer and novelist. Among engineering students studying in Canada, he is probably best known as the author of "The Ritual of the Calling of an Engineer" (often referred to as the Iron Ring Ceremony).

Kipling was not an engineer, but some of his poems and writings show that he held engineers in high regard. Among these poems is "The Sons of Martha", which was written in 1907. It celebrates the careful work done by engineers (and others) to provide for people's physical needs. In the poem, engineers are collectively referred to as the "Sons of Martha" (note that at the time the poem was written, few, if any, women were engineers).

In 1922, Kipling was asked to prepare an obligation and ceremony for students graduating from Canadian engineering schools. In response, he prepared "The Ritual of the Calling of an Engineer". A reading of "The Sons of Martha" was incorporated into the ceremony.